CAMERA BASED LANE DETECTION

Fazli Faruk OKUMUS (737548)

Mevluede TIGRE (737551)

TODAY'S AJANDA

Motivation

In recent years, autonomous vehicles have gained speed and importance with today’s technology.
Lane detection and lane-keeping are among the most important building blocks of autonomous driving.
To enable this, we proposed 2 different algorithms which are both of them based on image processing based

- Simple Algorithm
- Advanced Algorithm

Also, we examine the performance of state-of-the-art approaches for lane detection.

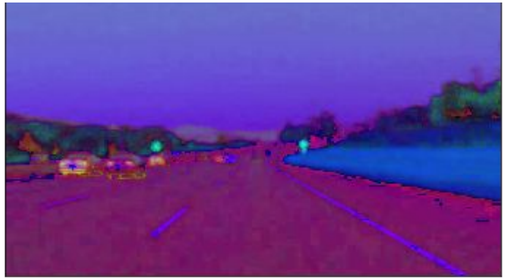
BACKGROUND

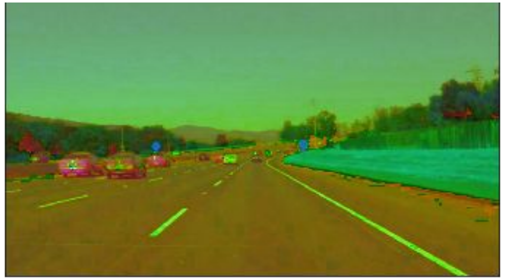
- Color Spaces
- Region of Interest(ROI)
- Perspective Transform

Color Spaces

- Colors can be represented by different methods.
- One of them is RGB(Red-Green-Blue) color space.
- But one color can be also represented by its Lightning, Saturation, etc. This can be very useful for different conditions and for extracting information and some parts from the image
- To extract lane lines, the characteristics of the lines have to be understood. Lines are white or yellow and most of the time they are evident and smooth.

RGB

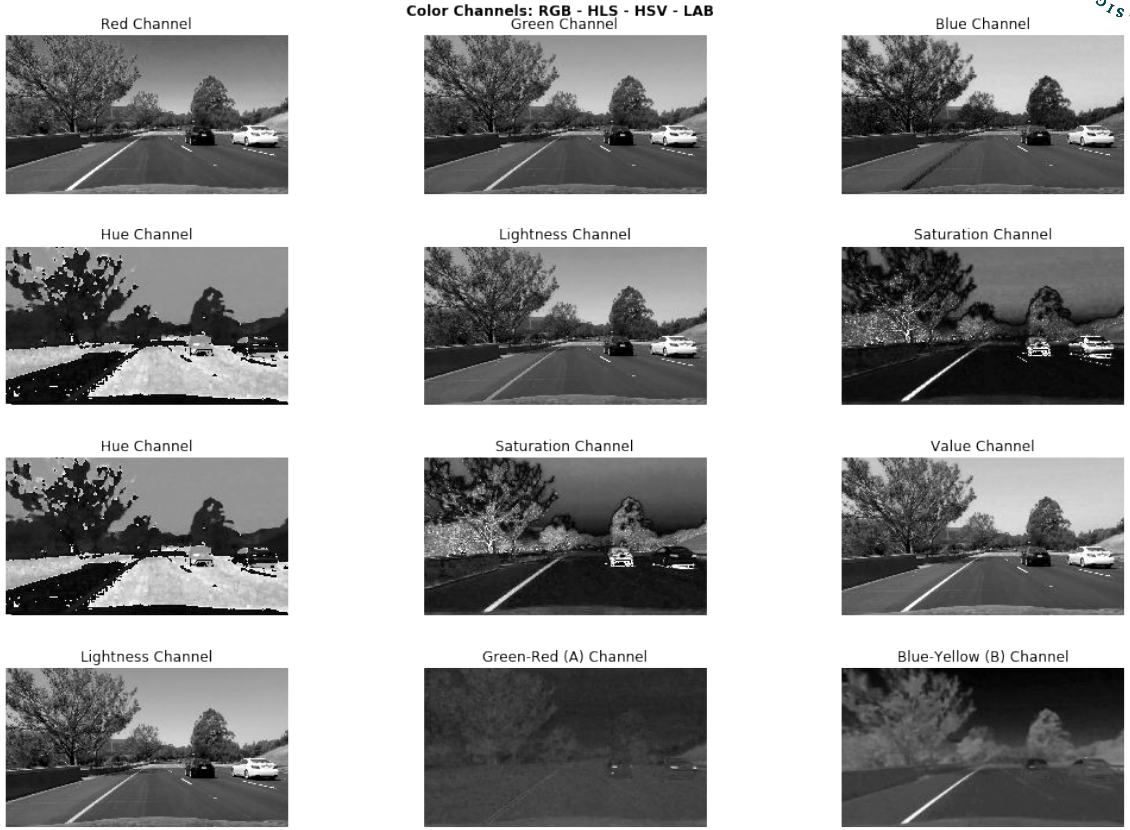

HSV


HSL

Background

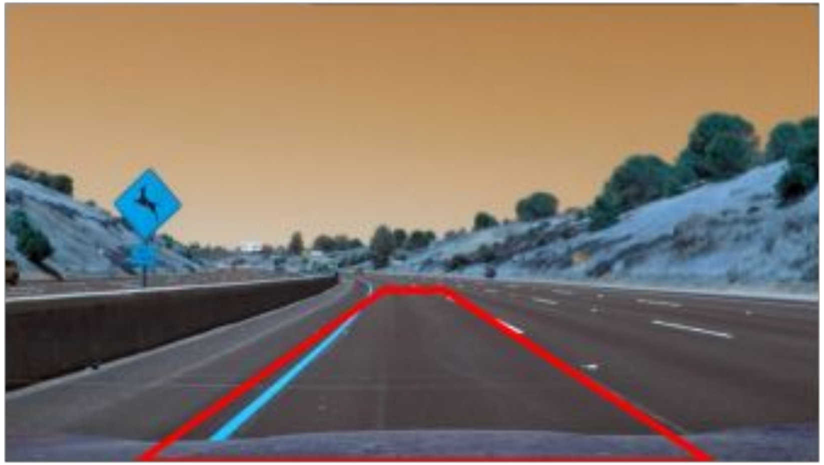
Test

- Saturation channels of HSV and HLS color spaces give very good results, despite the noisy output of the Hue channel.
- Also, the Lightness channel is giving grayish image and is good for identifying both yellow and white lane lines
- Thresholding values have been found for HLS color space to extract white and yellow colors.



Region of Interest (ROI)

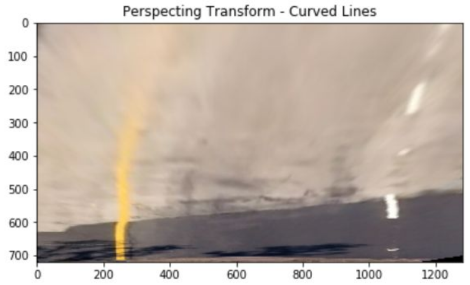
- A region of interest is a place on the image where interesting objects can be searched.
- It helps to narrow the interesting area.
- ROI coordinates were set by hand.

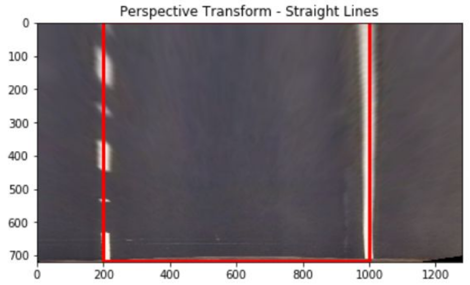


Perspective Transform

- ROI area in the 2D image is converted to a birds-eye view through a perspective transform technique.

- In the transformation, the location of each pixel will be transformed into a new position by using a transformation matrix.

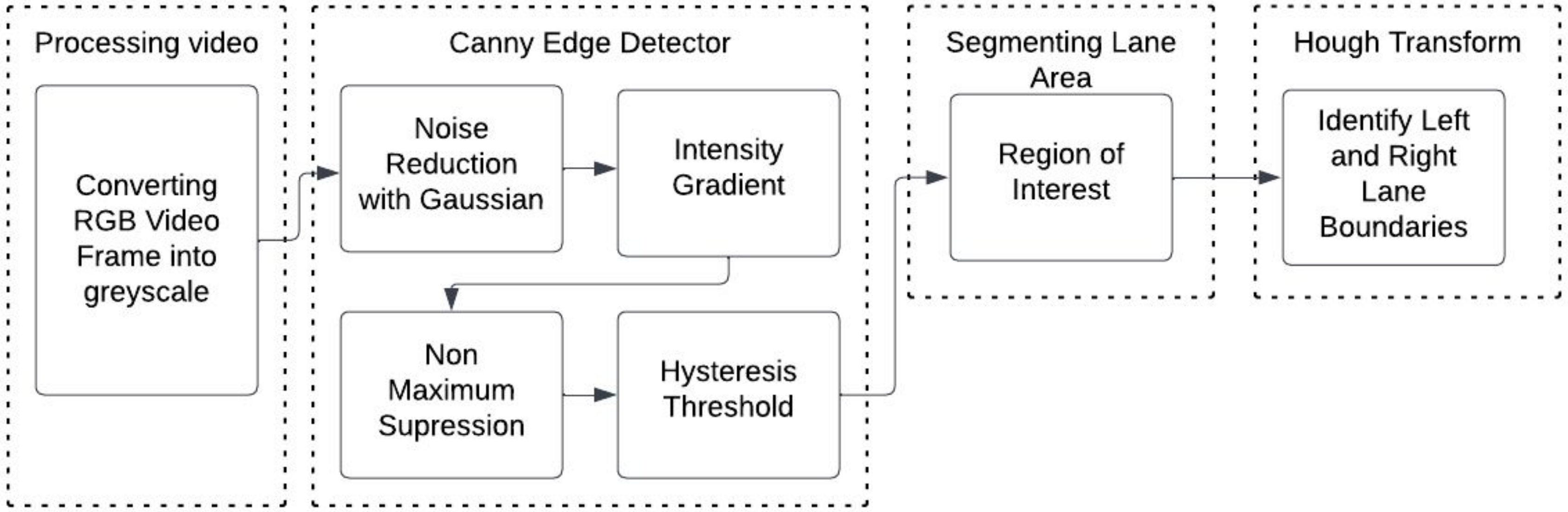




SIMPLE METHOD

- Canny Edge Detection
- Hough Transformation
- Optimizing and Displaying Lines

Simple Method



Canny Edge Detection

1 **Noise Reduction**

- Since edge detection is highly sensitive to noise in the image. To remove the noise in the image, we apply a Gaussian filter and get smoothened image.

2 **Intensity Gradient Calculation**

- After applying the Gaussian filter, the edge gradient and magnitude for each pixel are detected respectively.

$$\|\nabla I\|_2 = \sqrt{I_x^2 + I_y^2}$$

$$\Theta = \arctan \frac{I_y}{I_x},$$

3 **Non-maximum Suppression**

- To mitigate the thick edges, any unwanted pixels which may not constitute the edge are removed.

4 **Hysteresis Thresholding**

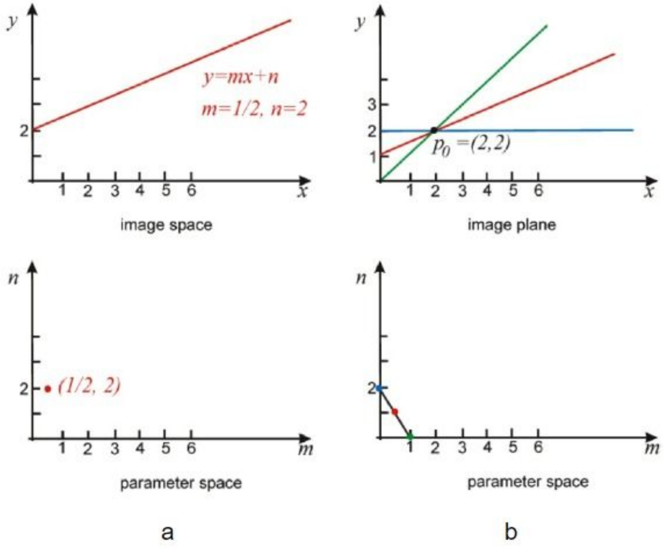
- Edges with an intensity gradient more than a high threshold are sure to edge and those below a low threshold are sure to be non-edges.

Hough Transformation

Hough transformation can be applied to the pixels found by Canny Edge Detection.

- Line in the image space can be expressed with two variables(m, n).
- A point in the image space can be represented with a line in parameter space because infinitely many lines can pass through this point.

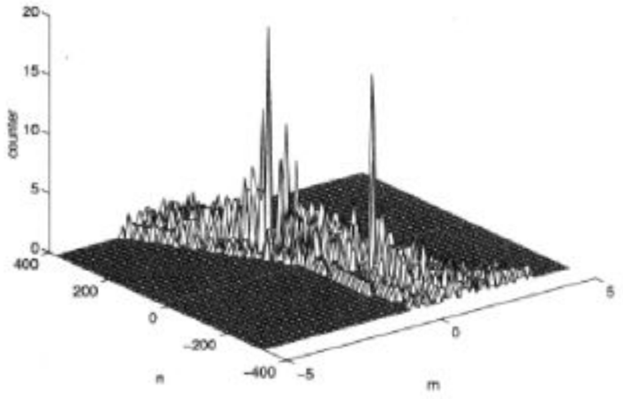
Line in the image space is defined via the intersection of lines in the parameter space that are associated with collinear points found by edge detection on line image space.



Hough Transformation

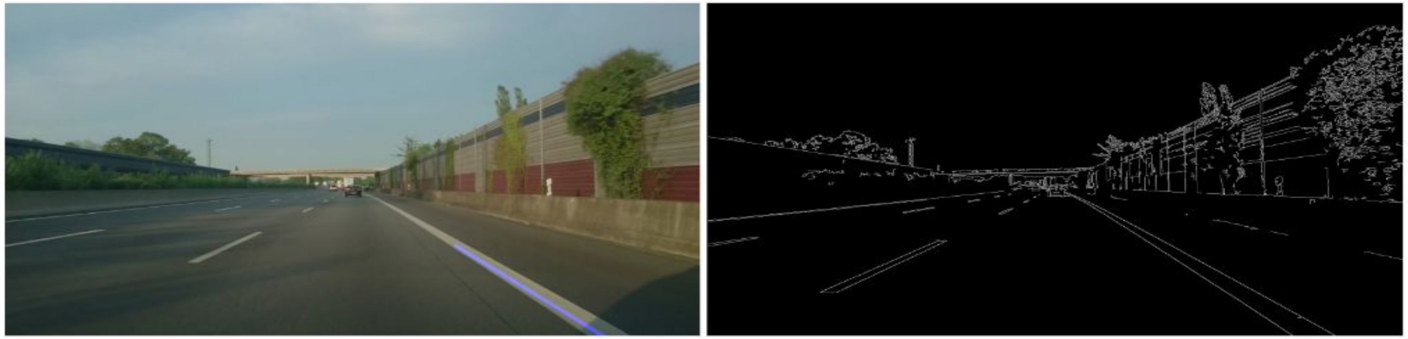
The problem turns into finding line intersections in parameter space as following steps :

- Divide the (m,n)-plane into a finite grid of cells
- Associate a counter cmn, initialized by zero, with each cell - this yields the accumulation matrix
- For each image edge point pi, we increment all counters on the corresponding line in parameter space.
- Counters at intersections will show higher counts than others.

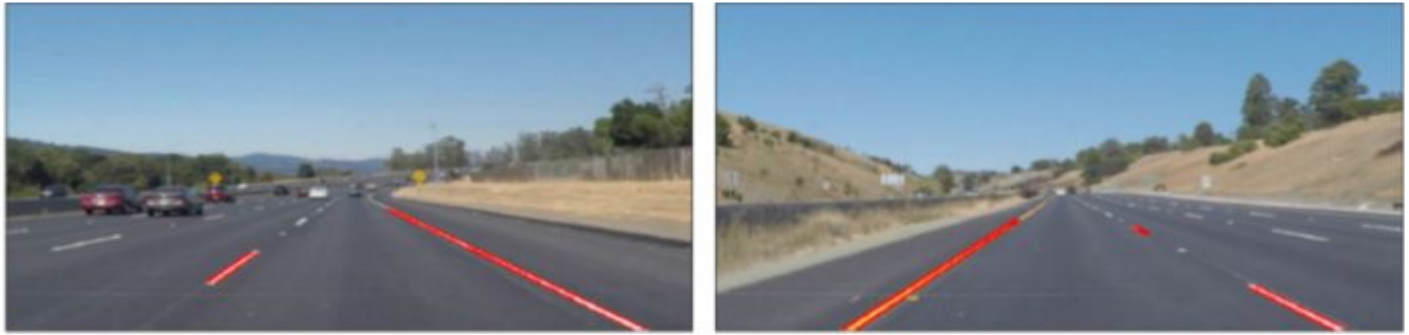


Simple Method

Output of Canny Edge Detector Algorithm

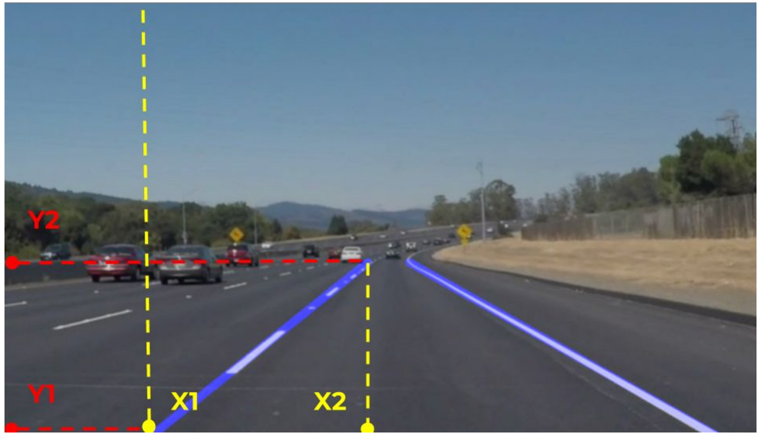


Output of Hough Tranfrom Algorithm



Optimizing Lines

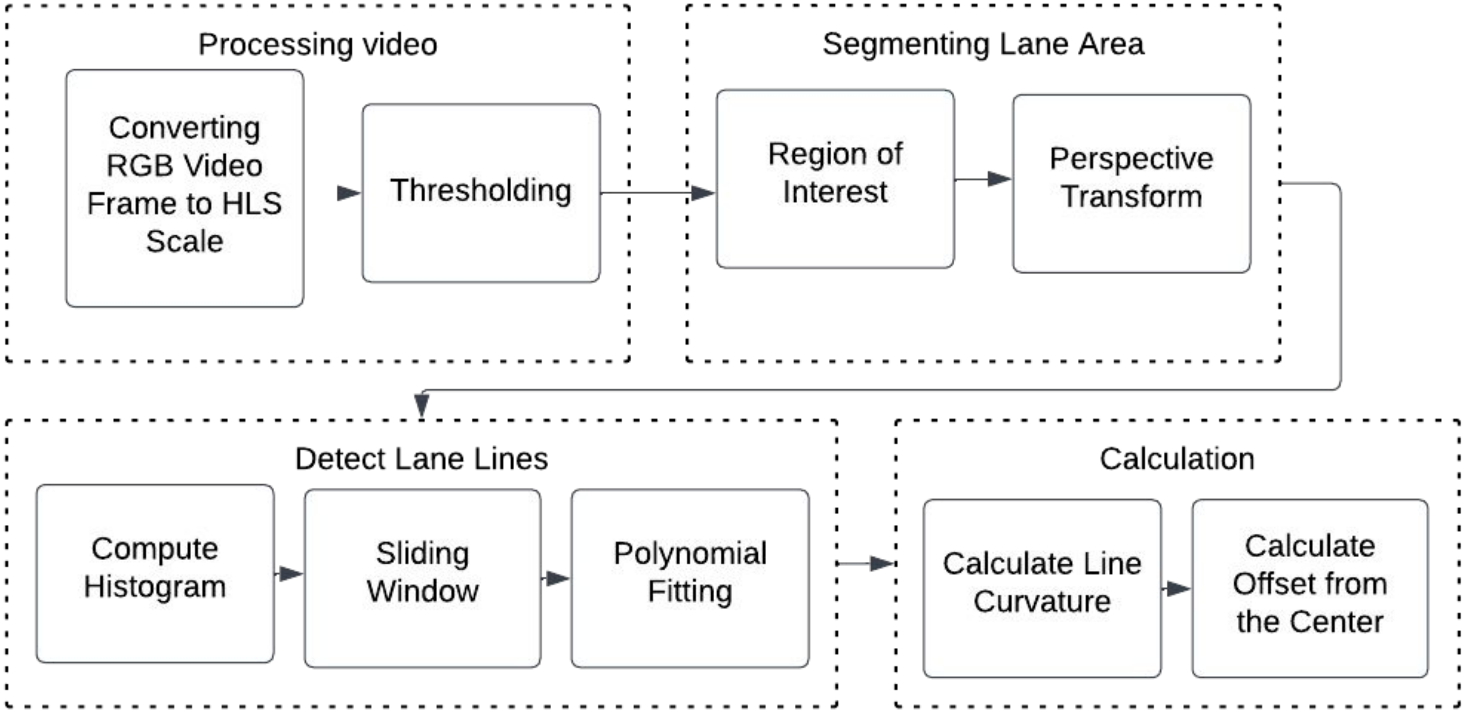
- Hough transformation gives dashed discontinued lines that depict edges of lines.
- Dashed lines are taken and categorized as belonging to the left or right.
- Lines that have a positive slope belong to the right side of the lane or are assigned to the left side of the lane.
- In the end, lines are calculated as shown in the Figure below. In this way, we can get a solid continuous left and right lines.



ADVANCED METHOD

- Thresholding
- Histogram and Sliding Window
- Line Curvature and Offset from the Center

Advanced Method



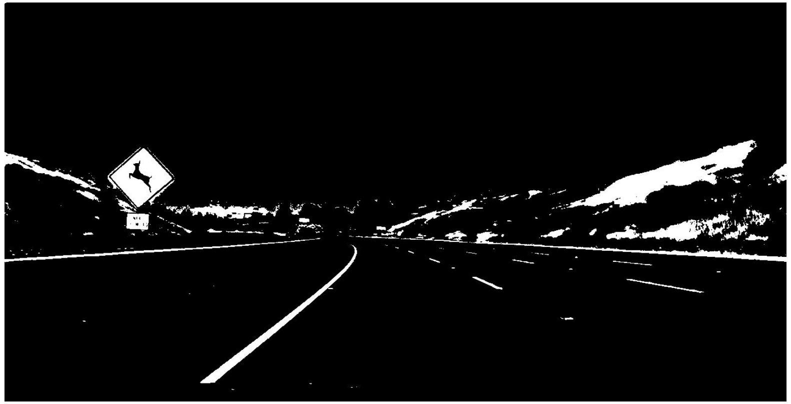
Thresholding

1 Color Thresholding

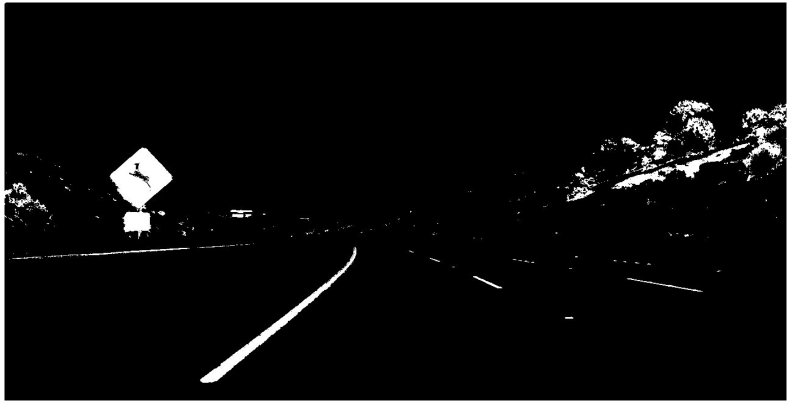
- The goal is to find the right thresholds on a given color channel to highlight the yellow and white lines of the lane.

.

- The Color Threshold module is used to:
  - Remove parts of the image outside a specified color range.
  - Used to detect objects of consistent color values.



(RGB)



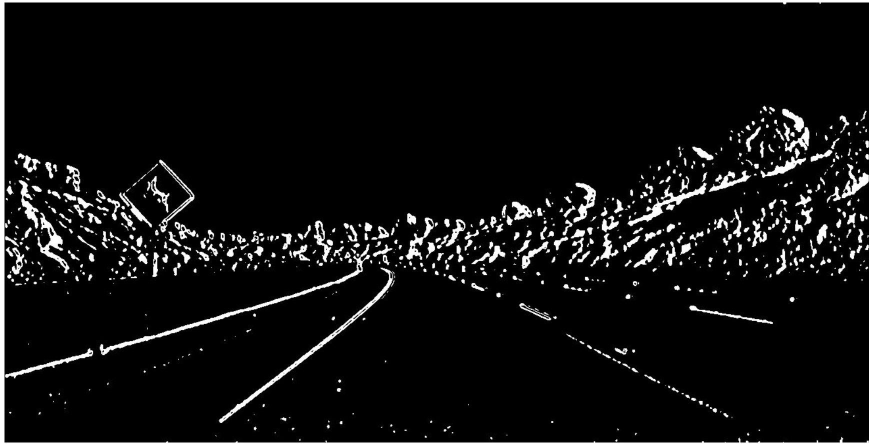
(HLS)

Advanced Method

2

Gradient Thresholding

- The Sobel operator is used to identify gradients, that is changes in color intensity in the image.
- Higher values represent strong gradients, and therefore sharp changes in color.
- Kernels are applied separately in each orientation (call these Ix and Iy).
- Then, these are combined together to find the absolute magnitude.

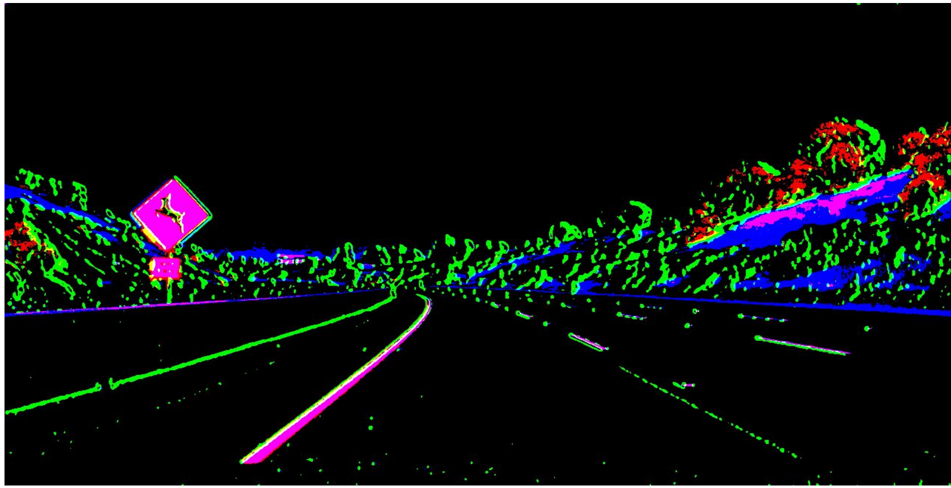


Advanced Method

3

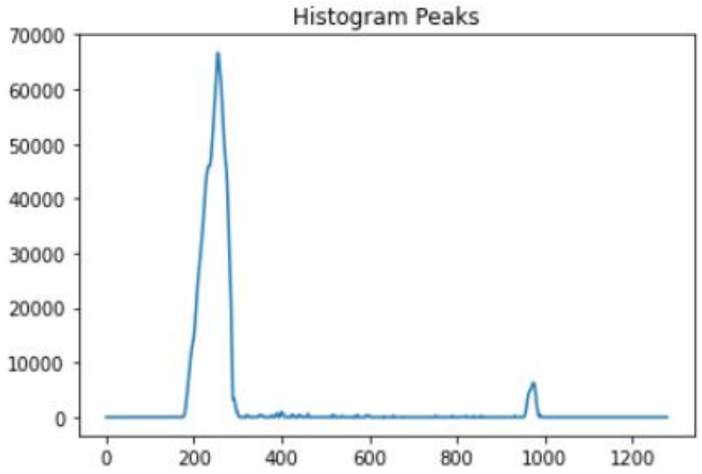
Combined Thresholding

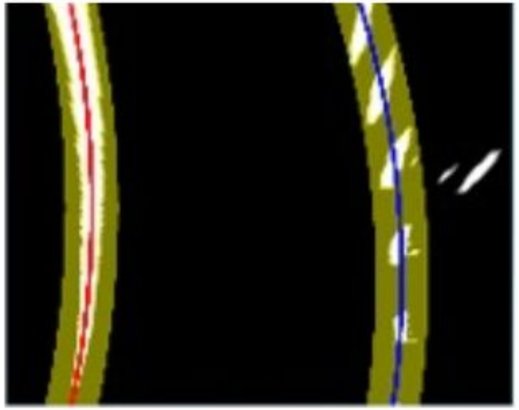
- The lane information is only taken from HLS thresholding.
- RGB color thresholding given blue color can be discarded to get rid of the noise.
- As a result, only a combination of the Color and Gradient thresholding gives the correct parameters to detect lanes in a robust manner.



Histogram

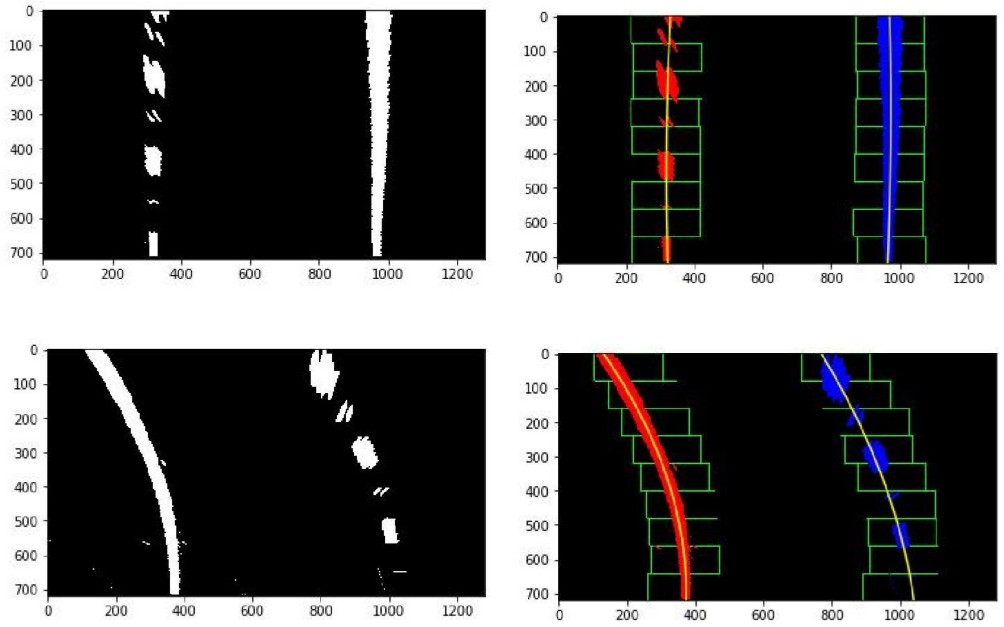
- The histogram of the Bird-eye view image has two peaks each representing one of the lane lines.
- The first peak represents the left line and the second one shows the right line.
- The histogram is only applied just on the bottom half of the image.





Sliding Window

- The sliding window technique takes these peaks as a starting point for searching the line pixels.
- Once it found pixels more than the predetermined threshold.
- It finds their average positions and takes this as a starting point for the next window.
- The search starts from the bottom and goes upwards.
- In the end, using these pixels, the algorithm fits a second-degree polynomial for each left and right lane line.



Calculate Line Curvature and Offset from the Center

- By using the location of the two detected lane lines found by sliding windows, calculate the curvature of the road and the car positions relative to the road can be calculated.

  Assumption : Camera is in the middle of the road and already calibrated.

$$\text{Radius of curvature} = \frac{\left[1 + \left(\frac{dy}{dx}\right)^2\right]^{3/2}}{\left|\frac{d^2y}{dx^2}\right|}$$

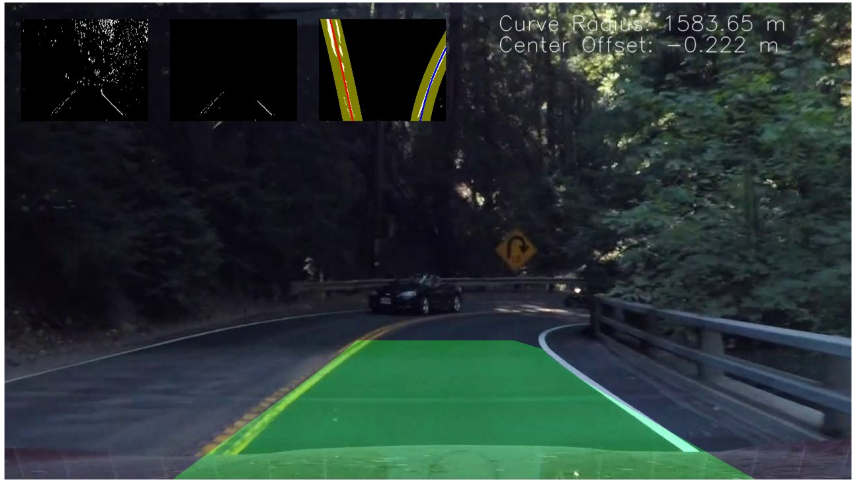
- On a straight lane the radius would be quite big, otherwise on a road has a high curvature, the radius would be small.
- Center point is calculated by finding the average coordinates for the left and right lines.

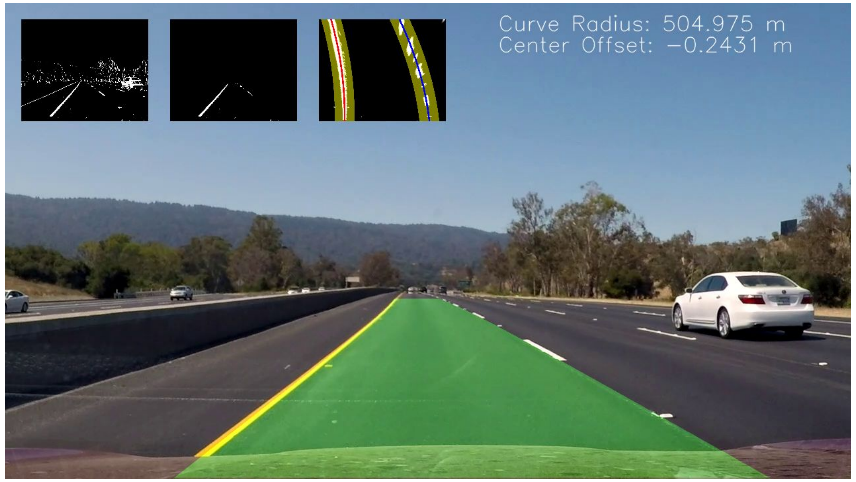
RESULTS AND COMPARISON

- Simple and Advanced Method Comparison
- Deep Learning Based Methods

SIMPLE VS ADVANCED COMPARISON

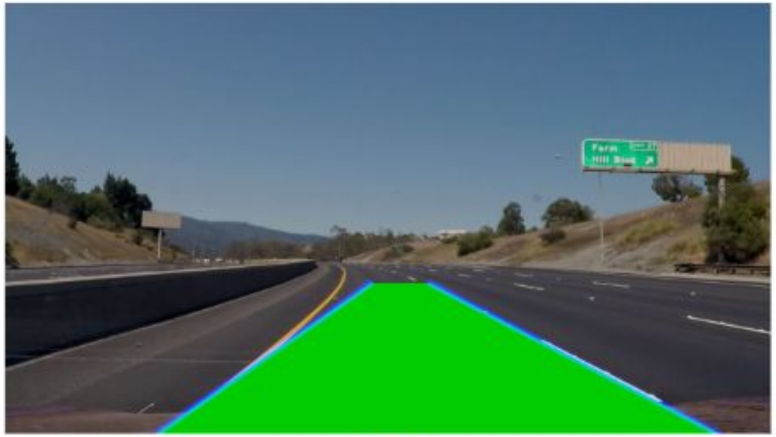
- The detection algorithms are tested in 3 different types of video in terms of process speed and the total number of correct detected frames.
- Two of the videos are in the highway conditions, lanes generally are visible.
- Lastly, the third video is in mountainous conditions. It has harsher conditions for the implemented detection algorithm like occlusions, absence of lane line, sudden and big brightness changes, and high curvatures

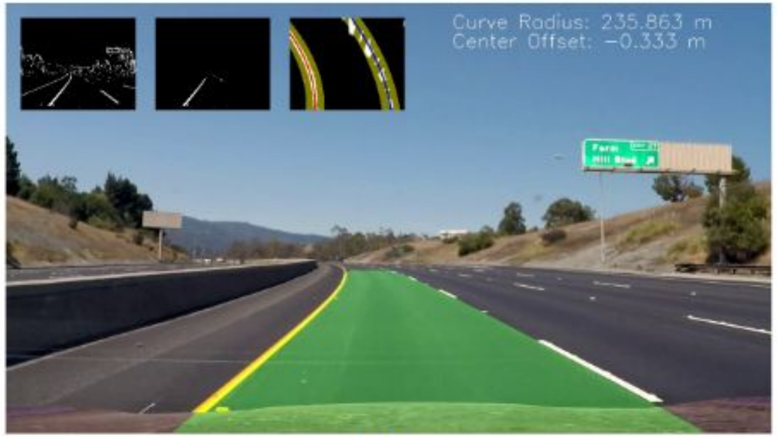




Simple and Advanced Methods Comparison

- The advanced algorithm is way better than the simple algorithm in terms of performance.
- Advanced one can cover lanes way better because it uses a second-order polynomial for each lane line, so can represent turns more decent.
- Also, we can find the curvature of the lane and car offset thanks to an advanced algorithm.

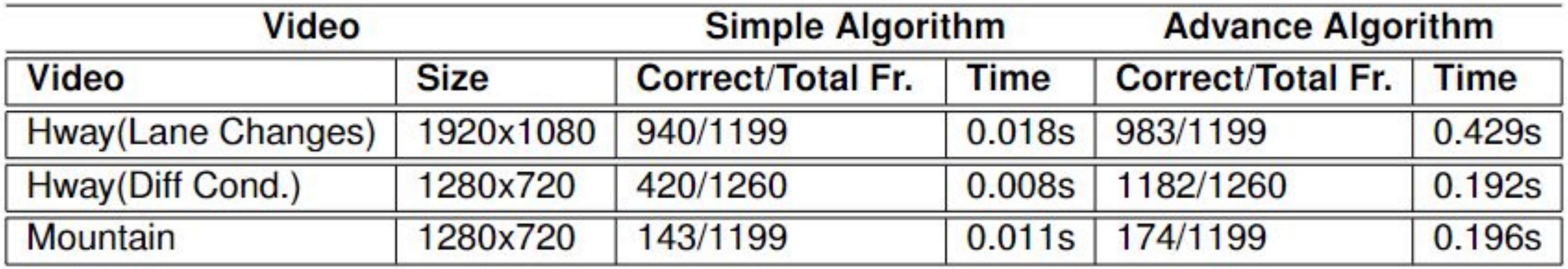

(a)



(b)

Results and Comparison

- In terms of working speed, a simple algorithm is pretty good, approximately can give 60 Hz information. The advanced algorithm can only give us 2-3 Hz information, so it can not be used for real-time operations.

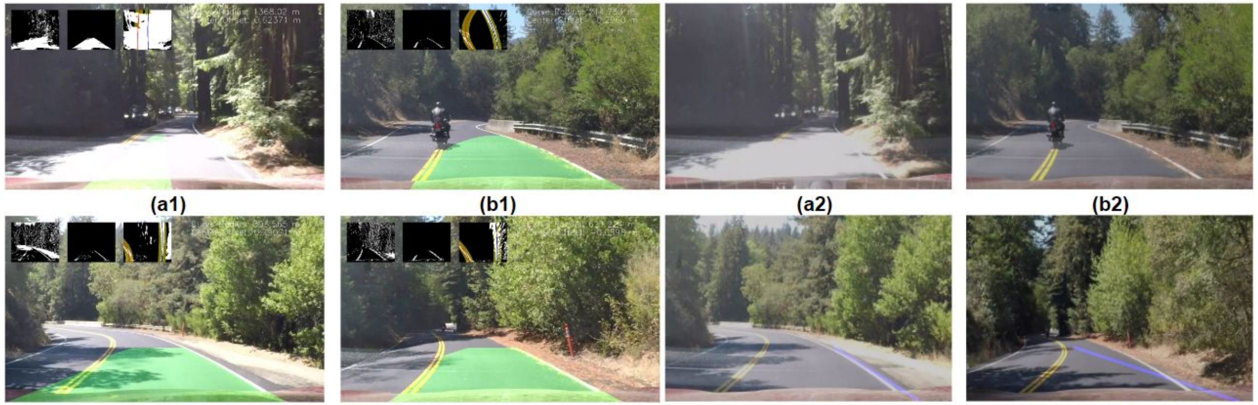


Video		Simple Algorithm		Advance Algorithm	
Video	Size	Correct/Total Fr.	Time	Correct/Total Fr.	Time
Hway(Lane Changes)	1920x1080	940/1199	0.018s	983/1199	0.429s
Hway(Diff Cond.)	1280x720	420/1260	0.008s	1182/1260	0.192s
Mountain	1280x720	143/1199	0.011s	174/1199	0.196s

Common Mistakes

Simple and advanced algorithms both basically failed on mountainous video because both algorithms are based on image processing and are highly physical, so lane lines need to be seen clearly.

- Sudden peak brightness changes make the algorithms completely blind.
- Tree shades or other vehicles can cause occlusions on the lane lines.
- Lastly, fallen leaves cover the lines which is also another problem.



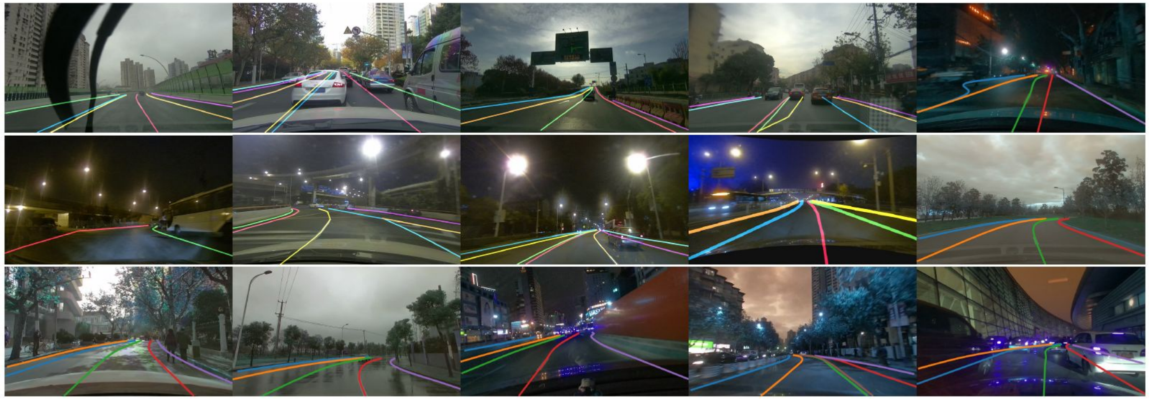
(a1) (b1) (a2) (b2)

(c1) (d1) (c2) (d2)

(1) Simple Method Results **(2)** Advanced Method Results

Deep Learning Based Methods

- For better and more reliable performance, methods based on Deep Learning could be an option.
- Well-known state-of-the-art approaches are CLRNet: Cross-Layer Refinement Network for Lane Detection which is trained on CULane and TuSimple and LLAMAS datasets.
- SCNN: Spatial CNN which is trained on CULane [8]. CULane and TuSimple and LLAMAS are the datasets that include images that have very hard conditions.



CLRNET vs SCNN

- In comparison, CLRNet adopts F1-measure as evaluation metric which is COCO metric. They are also introduce a new metric mF1 for better comparison, where F1@50, F1@55, · · · ,F1@95 are F1 metrics when IoU thresholds are 0.5, 0.55, · · · , 0.95.
- CLRNet performs both in terms of performance and also computation speed.
- Results are taken from the performance of both s-o-a approaches on the CULane dataset.
- SCNN can only give 7.5 Hz information. On the other hand, CLRNet gives approximately 100 FPS which is way better for real-time applications

$$F_1 = \frac{2 \times \text{ Precision } \times \text{ Recall}}{\text{Precision } + \text{ Recall}}$$

$$mF1 = (F1@50 + F1@55 + \cdots + F1@95)/10,$$

Method	mF1	F1@50	F1@75	FPS
SCNN	38.84	71.60	39.84	7.5
CLRNet	55.64	80.47	62.78	94

CONCLUSION

Conclusion

- It can clearly be seen that our approaches highly depend on the conditions of the road and lane lines should be visible all the time. Because of that, conditions like rain, snow, sudden brightness change, occlusion, and high curvature lower the accuracy of our algorithms.
- The implemented algorithms, especially the advanced algorithm, work well in highway conditions and when the lines are visible.
- But in case of needing more liability on any conditions, probably we need to move on with Deep learning-based approaches like CLRNet or SCNN.
- In terms of speed, our simple detection algorithm gives good results like the CLRNet. But, advanced detection algorithms could not keep up with them.
- To improve the speed of the advanced algorithm, we can introduce lane detection by the previous frame.
- To enhance the performance and speed of the advanced algorithm, different techniques rather than the sliding window technique can be the better solution, and also third-order B-spline can be obtained to cover highly curvature roads

THANKS!
Any Question ?